

REMARKS

Claims 13-16 and 18 are amended for purposes of expediting prosecution. Support for the amendment is provided by the example embodiments described in paragraph [0055] and shown in FIG. 8. Claims 1-4, 6-16, 18-19, 21-23 and 25-28 are pending in this application. In the discussion set forth below, Applicants do not acquiesce to any rejection or averment in the Office Action unless expressly stated. Reconsideration and allowance of the application are respectfully requested.

The Specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. The objection is thought to be improper since those skilled in the art would reasonably understand that the claimed "computer-readable medium" corresponds to the teachings of paragraph [0055], for example. However, claims 13-16 and 18 have been amended and the rejection should be withdrawn.

Claims 13-16 and 18 are understood to be directed to statutory subject matter under 35 USC §101 and the rejection is respectfully traversed. The rejection should be withdrawn because based on the guidance provided by MPEP §2106 II.A., the Office Action does not establish a *prima facie* case that the invention as a whole is directed solely to an abstract idea or to manipulation of abstract ideas or does not produce a useful result. However, for purposes of expediting prosecution, claims 13-16 and 18 are amended and the rejection should be withdrawn.

Claims 1-4, 6-16, 18-19, 21-23 and 25-28 are understood to be patentable under 35 USC §103(a) over "Sheshagiri" ("A Planner for Composing Services Described in DAML-S" to Sheshagiri et al.) in view of "Duftler" ("Web Services Invocation Framework" to Duftler et al.) The rejection is respectfully traversed because the Office Action does not show that all the limitations are suggested by the combination and does not provide a proper motivation for modifying the teachings of Sheshagiri with teachings of Duftler.

According to claim 1, the processor-implemented method for interfacing with a distributed computing service, comprises accessing an ontology specification describing messages of the distributed computing service; accessing a semantic interpretation specification that describes rules for semantically handling the messages, as specified in the ontology specification, with the distributed computing service; entering the

semantic interpretation specification into a rules engine adapted for providing processor executable procedures; obtaining a set of procedures from the rules engine for interacting with the distributed service based on the semantic interpretation specification; receiving a request for interfacing with the distributed service; and interfacing with the distributed computing service using the set of procedures in response to the request, wherein the interfacing comprises forming distributed computing service messages based on the ontology specification. These limitations are neither shown nor suggested by the Sheshagiri-Duftler combination.

The claimed invention enables a service that has been implemented as a given type of service, to present itself as a service of a different type. For example, a service that has been implemented as an Amazon Web service might need to present itself as an eBay Web service so that it can interact with some other application or service that can only interact with an eBay service. eBay would provide an eBay-service-specific ontology and an eBay-service-specific semantic interpretation specification. With the invention, an Amazon Web service accesses the eBay-service-specific ontology and the eBay-service-specific semantic interpretation specification and obtains the procedures for interacting with the eBay service from a rules engine. The Amazon Web service would then, in response to a request for interfacing with the eBay service, interface with the eBay service using the set of procedures and form messages based on the ontology specification, thereby presenting itself as the eBay service. That same Amazon Web service could just as easily take on a different semantic interpretation and present itself as some other service (e.g., an alibris Web Service).

Contrary to the assertions in the Office Action, Sheshagiri does not suggest the combination of accessing an ontology specification describing messages of the distributed computing service; accessing a semantic interpretation specification that describes rules for semantically handling the messages, as specified in the ontology specification, with the distributed computing service; entering the semantic interpretation specification into a rules engine adapted for providing processor executable procedures; and obtaining a set of procedures from the rules engine for interacting with the distributed service based on the semantic interpretation specification. Sheshagiri provides a DAML-S-specific solution to the problem of how to

guide the flow of interactions between services (Abstract). Sheshagiri addresses the problem of how a planner could compose services into a composite service (Abstract). As such, Sheshagiri describes a mechanism that converts DAML-S Service Model descriptions of services into Verb-Subject-Object triples that a reasoning engine could use to build a service conversation (Section 4). For example, Sheshagiri at the end of Section 4 describes how the tool would reason that in order for a given service to sell a book to a user, it must first support a login method, then get the user info, then look up the requested book, etc. Sheshagiri (at the end of Section 6) discusses how a generic description of a service may be provided. However, the intent is that this description would be generic enough to work with other services. Sheshagiri does not discuss the possibility that a service could take on any one of a number of interfaces, thereby presenting itself as more than one specific service by way of the claimed accessing of an ontology specification and a semantic interpretation specification, obtaining a set of procedures from the rules engine, and then interfacing with the distributed computing service using the set of procedures.

Duftler's goal is to provide a framework that provides a programmatic interface to a new Service (Abstract). As such, Duftler inputs WSDL service descriptions, which describe the structure of a Web Service (Section 3), as opposed to the implementation or the semantics. Duftler automatically generates a programmatic interface to be used by a human programmer when writing code to interface with the service (Sections 3 and 6). Duftler takes a description of a structure, and then provides an unpopulated complex object that can be incorporated into some piece of code (Section 4, page 6). The human programmer must still fill in the methods that supply parameter values. Duftler does not address the problem of how an application or service that implements one interface for a Web Service can dynamically implement a different interface. Thus, Duftler does not suggest the claimed interfacing with a distributed computing service using the set of procedure obtained from the rules engine.

The following paragraphs from the Detailed Description of the current application may be helpful in understanding the differences between the teachings of the Sheshagiri-Duftler combination and the claim limitations. Applicants recognize that the Examiner will not import teachings from the specification into the claims in order to

determine patentability. However, those skilled in the art will recognize that the invention recited in claim 1 is novel and non-obvious over the prior art, and Applicants repeat the following description in hopes of clarifying for the Examiner the significance of the claim limitations:

[0025] The descriptions provided by UDDI, WSDL, WSFL, and other specification standards can provide important facts about utilizing a Web service. However, these frameworks do not provide enough information to allow a program or application to discover and use the Web service without human intervention. In particular, the published specifications (e.g., WSDL descriptions) dictate the structure of the business documents involved in any particular transaction (e.g., using an XML schema description). However, current specifications do not address the semantics of the documents (e.g., that a certain field from document type 1 is equivalent to a given field from document type 2).

[0026] Ontologies help specify the semantics of document types. An ontology is a metadata schema that provides a formal, machine-processable, explicit specification of a set of terms. Ontology languages such as DAML+OIL and OWL offer standard mechanisms for structuring knowledge about entities in different domains in terms of classes, relationships, and inference rules that operate upon those classes and relationships. An ontology may define classes and class relationships, instances, slots/values, inheritance, constraints, relations between classes, and reasoning tasks.

[0027] However, even with WSDL, UDDI, and ontologies, human intervention is still required. Even if a client implements internal methods that can process and generate the documents compatible with a given Web service, a mapping between the client and service interfaces is still required. Published ontologies may allow services to specify the structure and vocabulary of the documents contained in Web service messages, but may not provide support for the exchange of methods and behaviors required to handle (e.g., interpret and produce) such messages. For example, a Web service might publish an interface that specifies that it expects the client to submit a ShippingAddress document (described by a ShippingAddress.xsd schema) at some point in the interaction. Even if the client implements an internal method that can generate a ShippingAddress document, the developer implementing the client must manually hook that method up to the interface. (That is, the service cannot automatically invoke that method from the client.) If the client implements methods that indirectly generate the document (e.g., method1 generates BillingAddress.xsd documents, and method2 converts BillingAddress.xsd documents into ShippingAddress.xsd documents), then manual intervention from the developer is even more necessary.

[0028] The need for human intervention makes Web services non-trivial to utilize. Even if the Web service can be understood and exploited by a non-technical user, an end-user application must often be programmed to interface with the Web service. A programmer is used to analyze and implement a programmatic Web service interface for the end-user application. Because this Web service interface is customized for the purposes of the end-user application, further intervention by a programmer is needed if the interface

changes. Similarly, a programmer may be needed to adapt the end-user application to exploit similar Web services that have different interfaces.

[0029] In order for a client to dynamically discover and implement a Web service, the published message ontologies can be associated with specifications for semantic interpretation. These semantic interpretation specifications express rules for handling the message using the published ontology. These semantic interpretation rules can be expressed, for example, using Java Expert System Shell (JESS) rules. In a simple example, the rules can identify a programmatic interface that enables the client to handle a certain document type. In a more sophisticated embodiment, a series of rule sets can enable the client to populate fields in a document (like filling out a form).

With this description those skilled in the art will recognize that the Sheshagiri-Duftler combination does not suggest the limitations of accessing an ontology specification, accessing a semantic interpretation specification, ... obtaining a set of procedures from the rules engine for interacting with the distributed service based on the semantic interpretation specification, ... and interfacing with the distributed computing service using the set of procedures.

The asserted motivation for combining Duftler with Sheshagiri is unsupported by evidence and improper. The Examiner asserted that "it would have been obvious ... to modify the invention of Sheshagiri to incorporate the features of Duftler ... because this allows services to be invoked in an abstract manner, independent of protocol bindings and their implementations and allows application code that uses Web Services to remain independent of the details of the use of those Web Services." There is no evidence presented that Sheshagiri does not already provide this capability or that Duftler's approach provides any appreciable benefit over the approach taught by Sheshagiri. For example, Sheshagiri teaches that "with WSIF, the user programs to the abstract service representations, instead of programming to a specific client-side implementation of a protocol such as SOAP." (Section 1, page 2). Thus, the asserted motivation is unsupported by evidence and improper.

Independent claims 7, 13, 19, 21, and 25 include limitations similar to those of claim 1 as discussed above. Claims 2-4 and 6 have claim 1 as a base claim; claims 8-12 have claim 7 as a base claim; claims 14-16 and 18 have claim 13 as a base claim; claims 22-23 have claim 21 as a base claim; and claims 26-28 have claim 25 as a base claim. Thus, the Sheshagiri-Duftler combination does not suggest the limitations of

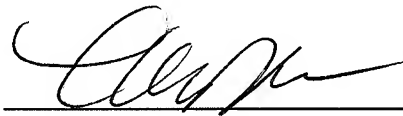
these claims for at least the reasons set forth above. The rejection of claims 1-4, 6-16, 18-19, 21-23 and 25-28 should be withdrawn because a *prima facie* case of obviousness has not been established.

Withdrawal of the rejections and reconsideration of the claims are respectfully requested in view of the amended claims and remarks set forth above. No extension of time is believed to be necessary for consideration of this response. However, if an extension of time is required, please consider this a petition for a sufficient number of months for consideration of this response. If there are any additional fees in connection with this response, please charge Deposit Account No. 50-0996 (HPCO.125PA).

Respectfully submitted,

CRAWFORD MAUNU PLLC
1150 Northland Drive, Suite 100
Saint Paul, MN 55120
(651) 686-6633

By: _____



Name: LeRoy D. Maunu

Reg. No.: 35,274